Extension of inverse-based dropping techniques for ILU preconditioners

A. Rafiei*, M. Bollhöfer †

September 21, 2010

Abstract

In this paper we present a safe and efficient way of dropping for those class of ILU factorizations that their factors are extracted as by-products of AINV process. The drop tolerance parameters of W and Z factors of AINV are selected the same as the drop tolerance parameters of L and U factors respectively. The infinity norms of the columns of W and Z are used to drop entries of L and U. The new dropping technique on L and U is based on monitoring the information of the inverses of L and U. Different dropping strategies for W and Z affect the efficiency of dropping for L and U.

Key words: right and left-looking RIF preconditioners, ILU factorization, inverse-based dropping technique
AMS Subject Classification : 65F10, 65F05.

1 Introduction

There are ILU [2, 4, 6, 9, 10, 11, 12] and AINV [1, 3, 7, 8] class of preconditioners to accelerate solving linear system of equations of the form:

$$Ax = b. \tag{1}$$

In the ILU class of preconditioners, the approximate factorization $A \approx LDU$ is being computed while in AINV calss of precondtioners, the approximate inverse factorization $A^{-1} \approx ZD^{-1}W^T$ is being generated. After analyzing the way of dropping and employing these two class of preconditioners to solve the preconditioned system of (1), it came to mind that how safe the way of dropping can be for ILU preconditioners?

Based on KIJ [10] and crout [11] versions of Gaussian elimination process, two ILU factorizations termed ILUSTAB [4, 6] and ILUC preconditioners has been presented in recent years. These two preconditioners belong to the class of ILU preconditioners that explicit or implicit dealing with the Schur-complement matrix is required. The characteristic of these two preconditioners provides the ability to employ a safe way of dropping technique for L and U factors. The strategy is based on monitoring effect of inverses of L and U in dropping. The main idea of this dropping technique is the fact that during the construction of the ILU preconditioners the L and U factors are being computed, but to apply the preconditioner to solve preconditioned system of (1), the L^{-1} and U^{-1} are needed. This way of dropping is called inverse-based dropping technique and needs to compute the approximation of the infinity norms $\|e_i^T L^{-1}\|_\infty$ and

*Department of Mathematics, Sabzevar Tarbiat Moallem University, Iran. Email: rafiei@sttu.ac.ir, rafiei.am@gmail.com.
†Institute computational Mathematics, Technische Universität Braunschweig, D-38106 Braunschweig, Germany. Email: m.bollhoefer@tu-bs.de

$\|e_i^T U^{-T}\|_\infty$ when matrices L^{-1} and U^{-T} are not completely available. To approximate the infinity norms, heuristic algorithms have been used in [4, 6, 11]. Although these heuristic algorithms play the main role for the efficiency of preconditioner, but nothing can be said about the precision of the approximations of the infinity norms, compared to their exact values, mathematically.

There is another type of ILU preconditioners, termed RIF [2, 9], that its L and U factors are extracted as by-products of AINV process. In this paper, we will extend the inverse-based dropping technique for both right and left-looking versions of this type of ILU preconditioners. Series of propositions, that are similar to the propositions presented in [5], will show that how one can skip computing approximations of the infinity norms $\|e_i^T L^{-1}\|_\infty$ and $\|e_i^T U^{-T}\|_\infty$ for the inverse-based dropping technique, when W and Z factors of AINV process are available.

In this paper, the notations $X_{:,i}$ and $X_{i,:}$ indicate the i-th column and i-th row of matrix X.

In sections two and three of this paper, inverse-based dropping technique for right and left-looking versions of RIF preconditioner are being presented. In section four, conclusions will be mentioned.

2 Inverse-based dropping technique for right-looking RIF preconditioner

The following algorithm will give an ILU preconditioner for a general nonsymmetric matrix. If the matrix is positive definite, then the preconditioner will exist, otherwise nothing can be said. The right-looking RIF preconditioner was first introduced for symmetric positive definite matrices. The preconditioner computed by Algorithm 1, is an extension of right-looking RIF preconditioner. Although it is different from right-looking RIF preconditioner, we still call it right-looking RIF.

Algorithm 1 (**extraction of** LDU **decomposition from right-looking AINV (SAINV) process**)

1. $w_i^{(0)} = e_i, \quad z_i^{(0)} = e_i \quad 1 \le i \le n$
2. **for** $i = 1$ to n **do**
3. $\quad v_i = Ae_i, \quad u_i = A^T e_i$ {not positive definite}
4. $\quad v_i = Aw_i^{(i-1)}, \quad u_i = A^T z_i^{(i-1)}$ {positive definite}
5. $\quad q_i^{(i-1)} = (w_i^{(i-1)})^T v_i, \quad p_i^{(i-1)} = (z_i^{(i-1)})^T u_i$
6. $\quad$ **for** $j = i+1$ to n **do**
7. $\quad\quad q_j^{(i-1)} = (w_j^{(i-1)})^T v_i, \quad p_j^{(i-1)} = (z_j^{(i-1)})^T u_i$
8. $\quad\quad L_{ji} = \frac{q_j^{(i-1)}}{q_i^{(i-1)}}, \quad U_{ij} = \frac{p_j^{(i-1)}}{p_i^{(i-1)}}$
9. $\quad\quad$ *apply a dropping rule to* L_{ji} *and to* U_{ij}
10. $\quad\quad w_j^{(i)} = w_j^{(i-1)} - (\frac{q_j^{(i-1)}}{q_i^{(i-1)}}) w_i^{(i-1)}, \quad z_j^{(i)} = z_j^{(i-1)} - (\frac{p_j^{(i-1)}}{p_i^{(i-1)}}) z_i^{(i-1)}$
11. $\quad\quad$ *for all* $l \le i$ *apply a dropping rule to* $w_{lj}^{(i)}$ *and to* $z_{lj}^{(i)}$
12. $\quad$ **end for**
13. **end for**
14. Let $d_{ii} = p_i^{(i-1)}$, for $1 \le i \le n$.
15. $L = (L_{ji})$, $U = (U_{ij})$ and $D = (d_{ii})$.

At step i of Algorithm 1, suppose that vectors $p^{(i)}$ and $q^{(i)}$ are defined as:

$$q^{(i)} = (0, \ldots, 0, \frac{q_{i+1}^{(i-1)}}{q_i^{(i-1)}}, \ldots, \frac{q_n^{(i-1)}}{q_i^{(i-1)}})^T, \qquad p^{(i)} = (0, \ldots, 0, \frac{p_{i+1}^{(i-1)}}{p_i^{(i-1)}}, \ldots, \frac{p_n^{(i-1)}}{p_i^{(i-1)}})^T.$$

Also, suppose that $W^{(i)}$ and $Z^{(i)}$ are the computed W and Z matrices at the end of step i and $W^{(i-1)}$ and $Z^{(i-1)}$ are the computed W and Z matrices at the end of step $i-1$ of Algorithm 1. We define matrices Q_i and P_i as:

$$Q_i = I - e_i(q^{(i)})^T, \qquad P_i = I - e_i(p^{(i)})^T,$$

where I is the identity matrix and e_i is the i-th column of this matrix. We also define G_i and T_i as the matrices that are produced by applying the dropping technique on entries $w_{tj}^{(i)}$ and $z_{tj}^{(i)}$ for $t < i; j > i$, at step i of Algorithm 1. One can show that the following two relations hold

$$W^{(i)} = W^{(i-1)}Q_i - G_i, \qquad Z^{(i)} = Z^{(i-1)}P_i - T_i. \tag{2}$$

Suppose that ε_Z and ε_W are the drop tolerance parameters for Z and W matrices, respectively. We present two different dropping strategies for W and Z matrices in Algorithm 1.

- **First strategy:** At each step i of Algorithm 1, for $t < i; j > i$, entries $z_{tj}^{(i)}$ and $w_{tj}^{(i)}$ are dropped when

$$|z_{tj}^{(i)}| \leq \varepsilon_Z, \qquad |w_{tj}^{(i)}| \leq \varepsilon_W. \tag{3}$$

  In this case, for $t < i; j > i$, just the entries $(T_i)_{tj}$ and $(G_i)_{tj}$ of matrices T_i and G_i are probably nonzero.

- **Second strategy:** At each step i of Algorithm 1, just for $t < i; j = i+1$, the entries $z_{tj}^{(i)}$ and $w_{tj}^{(i)}$ that satisfy criterion (3) are dropped. On the other hand, just the entries of the vectors $w_{i+1}^{(i)}$ and $z_{i+1}^{(i)}$ will be dropped. In this case, for $t < i$, only the entries $(T_i)_{t,i+1}$ and $(G_i)_{t,i+1}$ of matrices T_i and G_i are probably nonzero.

To implement Algorithm 1, as explained in [1], working with dynamic data structure is required. Exploiting first dropping strategy for W and Z matrices in this algorithm, will provide denser Z and W matrices. In this case, more memory will be needed and preconditioning time will be increased.

Proposition 2.1. *If in Algorithm 1, the first or second dropping strategy is used for W and Z matrices, then for $1 \leq l \leq i \leq n$ the two below relations hold*

$$G_iQ_l = G_i, \qquad T_iP_l = T_i.$$

Also, suppose that no dropping is applied for entries of matrices L and U. If matrices L_i^T and U_i are defined as:

$$L_i^T = Q_i^{-1}Q_{i-1}^{-1}\ldots Q_1^{-1}, \qquad U_i = P_i^{-1}P_{i-1}^{-1}\ldots P_1^{-1},$$

then

$$I - W^{(i)}L_i^T = \sum_{k=1}^{i} G_k, \qquad I - Z^{(i)}U_i = \sum_{k=1}^{i} T_k. \tag{4}$$

Proof: *See [5].*

Proposition 2.2. *Suppose that in Algorithm 1, entries of matrices L and U are not dropped . If in this algorithm, the first dropping strategy is used for W and Z matrices, then for $1 \leq i \leq j \leq n$ we have*

$$|(I - WL^T)_{ij}| \leq (j-i)\varepsilon_W, \qquad |(I - ZU)_{ij}| \leq (j-i)\varepsilon_Z.$$

If in this algorithm, the second dropping strategy is used for W and Z matrices, then for $1 \leq i \leq j \leq n$ we have

$$|(I - WL^T)_{ij}| \leq \varepsilon_W, \qquad |(I - ZU)_{ij}| \leq \varepsilon_Z.$$

Proof: *See* [5].

Until now, just the case of dropping for matrices W and Z has been surveyed in Algorithm 1. Although this will give robust right-looking RIF preconditioner, but the L and U factors will be so dense. The question that can be proposed here is that, besides applying the dropping technique for W and Z factors, what will be a safe dropping technique for L and U factors in Algorithm 1? The next proposition will reply this enquiry.

Proposition 2.3. *Suppose that in Algorithm 1, $\varepsilon_{L,W}$ is the drop tolerance parameter for matrices L and W and $\varepsilon_{U,Z}$ is the drop tolerance parameter for matrices U and Z. Also, suppose that in each step i of this algorithm, for $j > i$, entries L_{ji} and U_{ij} are dropped when*

$$|L_{ji}|\|W_{:,i}\|_\infty \leq \varepsilon_{L,W}, \qquad |U_{ij}|\|Z_{:,i}\|_\infty \leq \varepsilon_{U,Z}. \tag{5}$$

- *If in this algorithm, the first dropping strategy is used for W and Z matrices, then for $1 \leq i \leq j \leq n$ we have*

$$|(I - WL^T)_{ij}| \leq 2(j-i)\varepsilon_{L,W}, \qquad |(I - ZU)_{ij}| \leq 2(j-i)\varepsilon_{U,Z}. \tag{6}$$

- *If in this algorithm, the second dropping strategy is used for W and Z matrices, then for $1 \leq i \leq j \leq n$ we have*

$$|(I - WL^T)_{ij}| \leq (j-i+1)\varepsilon_{L,W}, \qquad |(I - ZU)_{ij}| \leq (j-i+1)\varepsilon_{U,Z}. \tag{7}$$

Proof: We just prove assertions (6) and (7) for L and W matrices. The proof for matrices U and Z will be the same.
Because of dropping, at step i of Algorithm 1, vector $\tilde{q}^{(i)}$ is computed instead of vector $q^{(i)}$. We define matrix $\tilde{Q}_i = I - e_i(\tilde{q}^{(i)})^T$. At the end of step i of Algorithm 1, we define matrix $\tilde{L}_i^T$ as:

$$\tilde{L}_i^T = \tilde{Q}_i^{-1}\tilde{Q}_{i-1}^{-1}\dots\tilde{Q}_1^{-1}.$$

For $k \leq i$, suppose that $\tilde{q}^{(k)} = q^{(k)} - f_k$. Therefore

$$\tilde{L}_i^T = L_i^T - \sum_{k=1}^{i} e_k f_k^T. \tag{8}$$

Relations (4) and (8) imply that

$$I - W^{(i)}\tilde{L}_i^T = \sum_{k=1}^{i} G_k + W^{(i)}\sum_{k=1}^{i} e_k f_k^T. \tag{9}$$

From relation (9) and the fact that $W = \sum_{k=1}^{n} w_k e_k^T$, one can show that

$$|e_i^T(I - W\tilde{L}^T)e_j| \leq |e_i^T(\sum_{k=1}^{n} G_k)e_j| + |e_i^T(\sum_{k=1}^{n} w_k f_k^T)e_j|.$$

Relation (5) indicates that

$$|e_i^T(\sum_{k=1}^{n} w_k f_k^T)e_j| \leq \sum_{i\leq k,j>k}^{n} |w_{ik}||f_k^T e_j| \leq \sum_{i\leq k,j>k}^{n} (\max_{t=1,\ldots,k} |w_{tk}|)|f_k^T e_j| \leq (j-i)\varepsilon_{L,W}.$$

If we use the first dropping strategy for matrix W, then $|e_i^T(\sum_{k=1}^{n} G_k)e_j| \leq (j-i)\varepsilon_{L,W}$ and if we use the second dropping strategy for this matrix, then $|e_i^T(\sum_{k=1}^{n} G_k)e_j| \leq \varepsilon_{L,W}$. By changing notation $\tilde{L}$ to L, the assertions (6) and (7) hold for matrices L and W. □

If we select the $\varepsilon_{L,W}$ and $\varepsilon_{U,Z}$ as the same drop tolerance parameters for matrices L, W and U, Z respectively, then the most efficient way of dropping in Algorithm 1, is exploiting the first dropping strategy for W and Z matrices and using criterion (5) to drop entries of L and U.

3 Inverse-based dropping technique for left-looking RIF preconditioner

The next algorithm will give an ILU preconditioner that we call it left-looking RIF preconditioner. Just for positive definite matrices there is a guarantee for existence of this preconditioner.

Algorithm 2 (extraction of LDU decomposition from left-looking AINV (SAINV) process)

1. **for** $i = 1$ to n **do**
2. $\quad w_i^{(0)} = e_i, \quad z_i^{(0)} = e_i$
3. $\quad$ **for** $j = 1$ to $i-1$ **do**
4. $\quad\quad q_i^{(j-1)} = (w_i^{(j-1)})^T A e_j, \quad p_i^{(j-1)} = (z_i^{(j-1)})^T A^T e_j$
5. $\quad\quad L_{ij} = \frac{q_i^{(j-1)}}{q_j^{(j-1)}}, \quad U_{ji} = \frac{p_i^{(j-1)}}{p_j^{(j-1)}}$
6. $\quad\quad$ *apply a dropping rule to* L_{ij} *and to* U_{ji}
7. $\quad\quad w_i^{(j)} = w_i^{(j-1)} - (\frac{q_i^{(j-1)}}{q_j^{(j-1)}})w_j^{(j-1)}, \quad z_i^{(j)} = z_i^{(j-1)} - (\frac{p_i^{(j-1)}}{p_j^{(j-1)}})z_j^{(j-1)}$
8. $\quad\quad$ *for all* $l \leq j$ *apply a dropping rule to* $w_{li}^{(j)}$ *and to* $z_{li}^{(j)}$
9. $\quad$ **end for**
10. $\quad q_i^{(i-1)} = (w_i^{(i-1)})^T A e_i, \quad p_i^{(i-1)} = (z_i^{(i-1)})^T A^T e_i$ {not positive definite}
11. $\quad q_i^{(i-1)} = (w_i^{(i-1)})^T A w_i^{(i-1)}, \quad p_i^{(i-1)} = (z_i^{(i-1)})^T A^T z_i^{(i-1)}$ {positive definite}
12. **end for**
13. Let $d_{ii} = p_i^{(i-1)}$, for $1 \leq i \leq n$.
14. $L = (L_{ji})$, $U = (U_{ij})$ and $D = (d_{ii})$.

At step i of Algorithm 2, suppose that vectors $q^{(i)}$ and $p^{(i)}$ are defined as:

$$q^{(i)} = (\frac{q_i^{(0)}}{q_1^{(0)}}, \ldots, \frac{q_i^{(i-2)}}{q_{i-1}^{(i-2)}}, 0, \ldots, 0)^T, \qquad p^{(i)} = (\frac{p_i^{(0)}}{p_1^{(0)}}, \ldots, \frac{p_i^{(i-2)}}{p_{i-1}^{(i-2)}}, 0, \ldots, 0)^T.$$

Suppose that $W^{(i)}$, $Z^{(i)}$ are the computed W and Z matrices at the end of step i and $W^{(i-1)}$ and $Z^{(i-1)}$ are the computed W and Z matrices at the end of step $i-1$ of Algorithm 2. We define matrices Q_i and P_i as:

$$Q_i = I - (q^{(i)})e_i^T, \qquad P_i = I - (p^{(i)})e_i^T.$$

We also consider G_i and T_i as the error matrices that are generated by applying the dropping technique on the entries $w_{li}^{(j)}$ and $z_{li}^{(j)}$ for $l < j; j < i$, at step i of Algorithm 2. It can be shown easily that still relation (2) holds between $W^{(i)}, W^{(i-1)}$ and $Z^{(i)}, Z^{(i-1)}$ matrices. Here we propose two different dropping strategies for W and Z matrices in Algorithm 2.

- **First strategy:** At step i of Algorithm 2, for $j < i$, the vectors $z_i^{(j)}$ and $w_i^{(j)}$ are computed and entries $z_{li}^{(j)}$ and $w_{li}^{(j)}$ for $l < j$, are dropped when

$$|z_{li}^{(j)}| \leq \varepsilon_Z, \qquad |w_{li}^{(j)}| \leq \varepsilon_W.$$

- **Second strategy:** At step i of Algorithm 2, at first, the vectors $z_i^{(i-1)}$ and $w_i^{(i-1)}$ are computed as:

$$z_i^{(i-1)} = z_i^{(0)} - \sum_{j=1}^{i-1} \frac{p_i^{(j-1)}}{p_j^{(j-1)}} z_j^{(j-1)}, \qquad w_i^{(i-1)} = w_i^{(0)} - \sum_{j=1}^{i-1} \frac{q_i^{(j-1)}}{q_j^{(j-1)}} w_j^{(j-1)},$$

  and then for $l < i$, the entries $z_{li}^{(i-1)}$ and $w_{li}^{(i-1)}$ are dropped when

$$|z_{li}^{(i-1)}| \leq \varepsilon_Z, \qquad |w_{li}^{(i-1)}| \leq \varepsilon_W.$$

In both dropping strategies, for $l < i$, just the entries $(G_i)_{li}$ and $(T_i)_{li}$ of matrices G_i and T_i are probably nonzero.

To implement Algorithm 2, as also explained in [1], working with static data structure is required. Applying first dropping strategy for W and Z matrices in this algorithm, needs less memory requirements and decreases the preconditioning time.

Both propositions 2.1 and 2.2 also hold for Algorithm 2. Although these two propositions will draw a pattern of dropping for this algorithm, but the most efficient way of dropping has been proposed in next proposition.

Proposition 3.1. *Suppose that in Algorithm 2, $\varepsilon_{L,W}$ is the drop tolerance parameter for matrices L and W and $\varepsilon_{U,Z}$ is the drop tolerance parameter for matrices U and Z. Also, suppose that in each step i of this algorithm, for $j < i$, entries L_{ij} and U_{ji} are dropped when*

$$|L_{ij}|\|W_{:,j}\|_\infty \leq \varepsilon_{L,W}, \qquad |U_{ji}|\|Z_{:,j}\|_\infty \leq \varepsilon_{U,Z}. \tag{10}$$

- *If in this algorithm, the first dropping strategy is used for W and Z matrices, then for $1 \leq j \leq i \leq n$ we have*

$$|(I - WL^T)_{ji}| \leq 2(i-j)\varepsilon_{L,W}, \qquad |(I - ZU)_{ji}| \leq 2(i-j)\varepsilon_{U,Z}.$$

- *If in this algorithm, the second dropping strategy is used for W and Z matrices, then for $1 \leq j \leq i \leq n$ we have*

$$|(I - WL^T)_{ji}| \leq (i-j+1)\varepsilon_{L,W}, \qquad |(I - ZU)_{ji}| \leq (i-j+1)\varepsilon_{U,Z}.$$

Proof: The proof of this proposition is the same as the proof of proposition 2.3. □

If we consider $\varepsilon_{L,W}$ and $\varepsilon_{U,Z}$ as the same drop tolerance parameters for matrices L, W and U, Z respectively, then exploiting the first dropping strategy for W and Z matrices and using criterion (10), is the most efficient way of dropping in Algorithm 2.

4 Conclusion

In this paper we showed that if we extract an LDU decomposition as by-product of right or left-looking AINV process, then it will be possible to apply a safe dropping technique for L and U factors. This dropping can be considered as an extension of inverse-based dropping technique. We proved that the quality of dropping technique for L and U factors, depends on the quality of the dropping technique we have used for W and Z factors of AINV process. The proposed dropping technique for the LDU decomposition, which is extracted from left-looking AINV process, is the most interesting point of this paper, since L and U are computed row-wise and column-wise respectively. This idea seems to be new, since for the inverse-based dropping techniques presented recently, L and U should be generated column-wise and row-wise respectively.

References

[1] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, SIAM J. Sci. Comput., 19(1998), 968-994.

[2] M. Benzi, M. Tůma, A robust incomplete factorization preconditioner for positive definite matrices, Numer. Linear Algebra Appl., 10(2003), 385-400.

[3] M. Benzi, J.K. Cullum, M. Tůma, Robust approximate inverse preconditioning for the conjugate gradient method, SIAM J. Sci. Comput., 22(2000), 1318-1332.

[4] M. Bollhöfer, A robust ILU based on monitoring the growth of the inverse factors, Linear Algebra Appl., 338(2001), 201-218.

[5] M. Bollhöfer, Y. Saad, On the relations between ILUs and factored approximate inverses, SIAM J. Matrix Anal. Appl., 24(2002), 219-237.

[6] M. Bollhöfer, A robust and efficient ILU that incorporates the growth of the inverse triangular factors, SIAM J. Sci. Comput., 25(2003), 86-103.

[7] S.A. Kharchenko, LYu. Kolotilina, AA. Nikishin, AYu.Yeremin, A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form, Numer. Linear Algebra Appl., 8(2001), 165-179.

[8] A. Rafiei, F. Toutounian, New breakdown-free variant of AINV method for nonsymmetric positive definite matrices, J. Comput. Appl. Math., 219(2008),72-80.

[9] A. Rafiei, F. Toutounian, Breakdown-free version of ILU factorization for nonsymmetric positive definite matrices, J. Comput. Appl. Math., 230(2009), 699-705.

[10] Y. Saad, Iterative methods for sparse linear systems, PWS publishing, Boston, 1996.

[11] Na Li, Y. Saad, E. Chow, Crout version of ILU for general sparse matrices, SIAM J. Sci. Comput., 25(2003), 716-728.

[12] Y. Saad, ILUT: A dual threshold incomplete ILU factorization, Numer. Linear Algebra Appl., 1(1994), 387-402.